第 38 卷 第 13 期
Vol.38 No.13
计 算 机 工 程
Computer Engineering
2012 年 7 月
July 2012

• 软件技术与数据库 •　　文章编号：1000—3428(2012)13—0064—03　　文献标识码：A　　中图分类号：TP311

基于向量的动态 XML 编码方法研究

冯少荣，陈天烁

(厦门大学计算机科学系，福建 厦门 361005)

摘　要：基于向量的动态可扩展标记语言(XML)编码方法计算简单，但不能对已删编码进行重用，严重影响 XML 更新效率。为此，利用 Stern-Brocot 树对中间向量计算进行改进，提出一种求解最短位长中间向量的多项式时间算法，对批量分配进行优化，从而提高向量编码的更新性能。实验结果证明改进的编码方法能较好地重用已删编码，适用于 XML 的频繁更新。

关键词：动态可扩展标记语言；向量编码；DDE 编码；中间向量；XML 编码

Research of Dynamic XML Coding Method Based on Vector

FENG Shao-rong, CHEN Tian-shuo

(Department of Computer Science, Xiamen University, Xiamen 361005, China)

【Abstract】eXtensible Markup Language(XML) coding method based on vector is simple but it can not reuse the deleted code, and it impacts the update efficiency for XML. An improved algorithm is proposed to calculate the intermediate vector with smallest size, which only takes polynomial time. Based on Stern-Brocot tree and modeling the problem as an integer programming problem, this algorithm improves the calculation of the intermediate vector and optimize the bulk allocation, so the vector encoding is more effective on update. Experimental result shows that the improved coding method can use the deleted code effectively and it is more adapted for XML update.

【Key words】dynamic eXtensible Markup Language(XML); vector coding; Dynamic Dewey(DDE) coding; middle vector; XML coding

DOI: 10.3969/j.issn.1000-3428.2012.13.018

1　概述

可扩展标记语言(eXtensible Markup Language, XML)作为新一代的 Web 数据表示和交换标准，在当前的互联网络和 IT 环境中扮演着重要角色。近年来，产生了各种 XML 相关索引和查询技术，大都是基于 XML 树的编码方法。编码方法首先应该能够高效地支持 XML 数据索引和查询，其次编码的长度既要短又要能支持动态编码更新[1]。

Dewey[2]编码和区间编码[3]是 2 种常用的静态编码方法，不能用于 XML 动态更新。CDBS[4](Compact Dynamic Binary String)和 QED[5](Quaternary Encoding)利用字符串代替整数进行编码，支持动态更新，但当倾斜插入时，会带来编码位长的快速增加，不可避免产生溢出问题。向量编码[6]利用二维向量代替一维的位字符串对 XML 进行编码，避免了倾斜插入时的溢出问题。DDE(Dynamic Dewey)[7]利用向量无限插入的性质，将 Dewey 编码扩展成更新支持的编码方法。

在上述主要的 XML 编码方法中，向量编码和 DDE 编码是 2 种基于向量的动态 XML 编码方法，两者都可以将 XML 更新转化为中间向量计算。已有方法直接利用相邻向量相加计算中间向量，计算简单，但是不能对已删节点编码进行重新利用，编码位长较长。较长的编码除了带来更多空间占用外，也会一定程度上降低查询效率。特别是删除和插入都发生频繁时，由于删除的节点编码得不到重新利用，导致编码长度增长更加明显。此外，有时也会带来一些信息的丢失，例如，从节点的 Dewey 编码可以方便地得到其所有祖先的编码，但这对 DDE 来说比较困难。基于此，本文提出改进的向量编码方法，其核心是计算最短位长中间向量。中间向量计算是一类整数规划问题，如果需要很大的时间花销才能得到最短位长的中间向量，改进没有意义。本文在追求更短位长的同时，对算法时间复杂度进行控制，可在 $O(\mathrm{lb}a)$时间复杂度内求得最短位长中间向量。同时，对批量分配进行优化，产生 n 个复杂度为 $O(n+\mathrm{lb}a)$的中间向量。

2　基于向量的 XML 编码方法

2.1　向量编码

向量编码形式是二维向量(m, n)，利用分数(m/n)进行大小判定。由于分数的无限可插性，因此使得向量编码可以支持 XML 动态更新。令 $V_M=V_L+V_R$，则 $V_L<V_M<V_R$，可以将 V_M 插入到 V_L 和 V_R 之间，而不修改原有编码。

编码长度是衡量编码方法好坏的重要指标。为讨论向量编码空间占用，引进距离和的定义。

定义 1(距离和 G) 向量 $V(x, y)$的距离和为 $G=x+y$。

为使编码所占空间尽可能小，中间向量的距离和要尽可能小。利用 $V_L<V_M=V_L+V_R<V_R$ 可以方便地计算中间向量，但距离和较大。例如，往(2,3)和(5,4)插入新向量，算法得到 $V_M=(7,7)$，远大于距离和最小的向量(1,1)。当 XML 只发生插入更新时，编码效率仍然较高，但当删除和插入都频繁发生时，编码增长较快。本文对向量编码提出改进，使其更适合于频繁更新。

2.2　Dewey 编码

Dewey 编码形式是多维向量，DDE 利用向量相加的性

基金项目：国家自然科学基金资助项目(50604012)

作者简介：冯少荣(1964－)，男，副教授，主研方向：XML 数据库技术，数据挖掘；陈天烁，硕士研究生

收稿日期：2012-02-01　**E-mail：**ailands@163.com

质，将 Dewey 编码扩展成更新支持的编码方法。当静态编码时，DDE 和 Dewey 编码一样，当动态插入时，只需往相邻的编码$(a_1a_2\cdots a_n)$、$(b_1b_2\cdots b_n)$间插入新编码$((a_0+b_0)(a_1+b_1)\cdots(a_n+b_n))$就可。其序关系判断如下：

$\boldsymbol{A}$: $(a_1a_2\cdots a_m)$，$\boldsymbol{B}$: $(b_1b_2\cdots b_n)$，$\boldsymbol{A}<\boldsymbol{B}$ 要满足以下条件：

(1)$m\leqslant n$ 且 $a_1/b_1=a_2/b_2=\cdots=a_m/b_m$；

(2)$k\leqslant \min(m, n)$，满足 $a_1/b_1=a_2/b_2=\cdots=a_{k-1}/b_{k-1}$，且 $a_k\times b_1<b_k\times a_1$。

注意到，往相邻的编码$(a_1a_2\cdots a_n)$、$(b_1b_2\cdots b_n)$之间插入新编码，在新编码$(c_1c_2\cdots c_n)$中，首尾分量 c_1 和 c_n 只需满足 $a_1/a_n>c_1/c_n>b_1/b_2$，当 c_1 和 c_n 确定后，$c_2c_3\cdots c_{n-1}$ 通过等式 $a_1/c_1=a_2/c_2=\cdots=a_{n-1}/c_{n-1}$ 唯一确定。可以看出，DDE 形式上是一个多维向量，但当动态插入时，需考虑的维度为首尾 2 维，即找到中间向量 $C(c_1,c_n)$，满足 $\boldsymbol{A}>\boldsymbol{C}>\boldsymbol{B}$，其中，$\boldsymbol{A}=(a_1,a_n)$；$\boldsymbol{B}=(b_1,b_n)$。

由此可见，DDE 可以转化为二维向量进行讨论。和二维向量编码一样，通过找到具有最小距离和的新向量对 DDE 进行改进。

3　改进的向量编码方法

向量$(k\times m, k\times n)$与(m,n)代表同一个分数，为使编码长度降低，向量(m,n)应具最简形式，即满足 m、n 不可约分。已有的向量编码不能保证新向量的 2 个分量不可约分，如果通过额外化简新向量进行优化，代价较大。改进的向量编码直接在所有不可约分数中找新向量。

3.1　问题建模

计算介于已有向量(m_1,n_1)、(m_2,n_2)最小位长的中间向量(x,y)是一类简单的整数规划问题，可建模为：求 $x+y$ 最小并满足 $m_1y-n_1x<0$；$m_2y-n_2x>0$；$x,y>0$ 且 x,y 是整数。

整数规划是 NP 问题，计算最小位长的中间向量是简单的整数规划问题，可在多项式时间内求解，基于 Stern-brocot 树，其复杂度为 $O(\text{lb}a)$，其中，$a=\max[\min(m_1,n_1),\min(m_2,n_2)]$。

3.2　Stern-Brocot 树

德国数学家 Stern R L 和法国钟表匠 Brocot D E 提出一种构造所有不可约分数的方法，即 Stern-Brocot 树[8](见图 1)。构造这棵树的主要思想是从向量(0,1)和(1,0)开始，新向量 $\boldsymbol{V}$ 是由其上的 2 个向量相加得到，即 $\boldsymbol{V}=\boldsymbol{startV}+\boldsymbol{endV}$，其中，$\boldsymbol{startV}$ 是其左边的上面最接近祖先；$\boldsymbol{endV}$ 是其右边的上面最接近祖先。

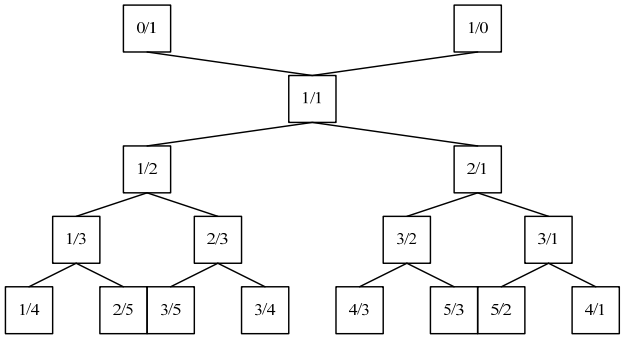

图 1　层次为 4 的 Stern-Brocot 树

向量在树上的路径可用二进制串表示，用“0”、“1”分别表示路径上的左右分支，其中，根节点的路径定义为二进制串“1”。特别的，向量(0,1)和(1,0)用二进制串“0”表示，可以方便地统一各种操作。向量 $\boldsymbol{V}(m, n)$与路径二进制串 S 对应。以下讨论其转化算法，应用中经常利用到 $\boldsymbol{V}$ 所对应的 $\boldsymbol{startV}$、$\boldsymbol{endV}$，算法同时实现 $\boldsymbol{V}$ 对应的 $\boldsymbol{startV}$。

将 S 转化为 $\boldsymbol{V}$ 相对简单，算法 1 是其实现。

算法 1 Procedure STOV(S, $\boldsymbol{V}$, $\boldsymbol{startV}$)

输入　S

输出　S 对应的 $\boldsymbol{V}$、$\boldsymbol{startV}$

```
1. V←(1,1)  startV←(0,1)
2. S<<1    // S<<n 表示往左移掉 n 位
3. while  S!=NULL  do
4.   if  S 开头最多有 n 个连续的 0  then
5.       S<<n
6.       V←V+n×startV
7.   else  if  S 开头最多有 n 个连续的 1  then
8.       S<<n
9.       endV←V–startV
10.      V←V+n×startV
11.      startV←V–endV
```

文献[8]给出了 $\boldsymbol{V}$ 转化为 S 的 $O(\max(m, n))$算法。

算法 2 Procedure SimpleVTOS($\boldsymbol{V}$)

输入　$\boldsymbol{V}(m, n)$

输出　$\boldsymbol{V}$ 对应的 S

```
1. S←1
2. While  m! = n  do
3. if  m<n  then  S←S⊕0  n←n–m// ⊕ 表示连接操作
4.            else  S←S⊕1  m←m–n
```

算法 3 对算法 2 进行改进，将减法转化为除法。

算法 3 Procedure VTOS($\boldsymbol{V}$, S, $\boldsymbol{startV}$)

输入　$\boldsymbol{V}(m, n)$

输出　$\boldsymbol{V}$ 对应的 S、$\boldsymbol{startV}$

```
1. if  m==0  then  S←0  startV←(1,0)  return
2. if  n==0  then  S←0  startV←(1, –1)  return
3. S←1  startV←(0,1)  V_T←(1,1)
4. while  m! =0&&n! =0  do
5.   if  m<n
6.     then  n←n % m  S←S⊕0^(n/m)  //0^i 表示 i 个 0
7.           V_T←V_T +(n/m)× startV
8.      else  m←m % n  S←S⊕1^(m/n)
9.           startV←V_T – startV  V_T←V_T +( m/n)× startV
10.  End
11. S>>1 // S>>n 表示往右移掉 n 位
```

算法 3 类似辗转相除，算法 1 是其反过程，和辗转相除法一样，复杂度为 $O(\log(\min(m, n)))$。

3.3　最小位数中间向量

定义 2(层次 i) Stern-Brocot 树根节点所在层次定义为 1，每下降一层，i 值加 1。

定义 3(位长 $size$) 向量 $\boldsymbol{V}$ 所对应二进制串 S 的位长，即 $\boldsymbol{V}$ 所在的层次，特别的，$size$(‘0’)=0。

当 2 个向量位于同一层时，介于这 2 个向量之间最短位长的中间向量就是其最近共同祖先，转化为路径二进制串表示时，就是这 2 个二进制串的最长公共前缀。当这 2 个向量不在同一层时，可转化为等价的同层向量进行求解。设完全二叉树的层次为 i，中序遍历该完全二叉树，则非叶节点的前驱和后继都是叶子节点，特别的，0 的后继前驱是中序遍历的首尾节点。利用前驱和后继可将非叶节点转化为叶子节点进行计算。

设向量 $\boldsymbol{V}_L$、$\boldsymbol{V}_R$ 所在层次最大为 i，将层次设定为 i+1，则 $\boldsymbol{V}_L$、$\boldsymbol{V}_R$ 都是非叶节点，且 $\boldsymbol{V}_L$ 的后继向量 $\boldsymbol{V}_L$’和 $\boldsymbol{V}_R$ 的前驱向量 $\boldsymbol{V}_R$’满足 $\boldsymbol{V}_L<\boldsymbol{V}_L'\leqslant\boldsymbol{V}_R'<\boldsymbol{V}_R$，由于 $\boldsymbol{V}_L$、$\boldsymbol{V}_R$ 自身不可能满足

V_M 定义，因此 V_M 只能存在于 V_L 的后继 V_L'与 V_R 的前驱 V_R'之间，求介于 V_L、V_R 间的 V_M，等价于求介于 V_L'、V_R'间的 V_M，而 V_L'、V_R'都是叶子节点，满足两叶子节点的 V_M 就是其最近共同祖先，其实现如算法 4 所示。时间复杂度取决于向量路径的获得，即 $O(\text{lb}a)$，其中，$a=\max[\min(V_L\cdot m, V_L\cdot n), \min(V_R\cdot m, V_R\cdot n)]$。

算法 4 *VectorWithSmallestSize*(V_L,V_R)

输入 $V_L<V_R$

输出 V_M

1. S_L←VTOS(V_L)
2. S_R ←VTOS(V_R)
3. S_M←StringWithSmallestSize(S_L,S_R)
4. V_M ←STOV(S_M)

Procedure *StringWithSmallestSize*(S_L,S_R)

1. i←1+max(size(S_L),size(S_R))
2. S_L←$S_L\oplus 10^{\text{i-size(s)-1}}$ //求 S_L 的后继
3. if S_R=0 then S_R←1^i //求 S_R 的前驱
4. else S_R←$S_R\oplus 01^{\text{i-size(s)-1}}$
5. S_M←S_L 和 S_R 的最长公共前缀
6. return S_M

3.4 批量分配

在 XML 应用中，数据在某个时间段批量插入是比较常见的，批量插入转化为求介于 2 个向量之间 n 个新向量。算法 5 递归获得 n 个向量，先产生 V_L、V_R 的中间向量 V_M，接着产生 V_L、V_M 的中间向量，V_M 和 V_R 的中间向量，直到产生 n 个向量为止。时间复杂度为 $O(n\times\text{lb}a)$。

算法 5 *BatchAllocate*(V_L, V_R, n)

输入 n、$V_L<V_R$

输出 n 个介于 V_L、V_R 间的新向量

1. Arr[0,n+1] //申请数组，n 个有效编码保存在 Arr[1,n]
2. Arr[0]←V_L Arr[n+1]←V_R
3. SubEncoding(Arr,0,n+1)
4. return Arr[1,n]

Procedure *SubEncoding*(*Arr*,l,*r*)

/*递归函数，*Arr* 是数组，*l*、*r* 是数组左右下标*/

1. if l +1> = r return
2. m←round((l+r)/2)
3. Arr[m]←VectorWithSmallestSize(Arr[l], Arr[r])
4. SubEncoding(Arr, l, m)
5. SubEncoding(Arr, m, r)

3.5 批量分配的算法优化

当向量 V_L、V_R 具有祖先后代关系，可简化中间向量 V_M 的计算。先设 V_L 是 V_R 的祖先节点，这时 V_R 位于 V_L 左子树中，利用 S_L、S_R 求得 V_M 路径 S_M，其形式是 $S_L\oplus 10^i$，可直接求得 V_M=*start*V_L+$i\times V_L$。V_R 是 V_L 的祖先节点时，可同理求得。当 V_L、V_R 不具有祖先后代关系，利用算法 5 求得 V_M，此后 V_L、V_M 与 V_M、V_R 都有祖先后代关系。可见，计算第 1 个中间向量需要 $O(\text{lb}a)$，剩下 $n-1$ 个都可以在 $O(1)$内得到，优化后的批量分配算法时间复杂度为 $O(n+\text{lb}a)$，如算法 6 所示。

算法 6 *BatchAllocatePlus*(V_L, V_R, n)

输入 n、$V_L<V_R$

输出 n 个介于 V_L、V_R 间的新向量

1. Arr[0,n+1] //n 个有效编码保存在 Arr[1,n]
2. Arr[0]←V_L Arr[n+1]←V_R
3. VTOS(V_L,S_L,startV_L)
4. VTOS(V_R,S_R,startV_R)
5. S_M←StringWithSmallestSize(S_L,S_R)
6. STOV(S_M,V_M,startV_M)
7. m←round((l+r)/2)
8. Arr[m]←V_M
9. SubEncoding(Arr,0,m,startV_L,startV_M,S_L,S_M)
10. SubEncoding(Arr,m,n+1,startV_M,startV_R,S_M,S_R)
11. return Arr[1,n]

Procedure *SubEncoding*(*Arr*,*l*,*r*,*start*V_L,*start*V_R,S_L,S_R)

/*递归函数，*Arr* 是数组，*l*、*r* 是数组左右下标，向量 ***Arr***[*l*]对应 *start*V_L、S_L，向量 ***Arr***[*r*]对应 *start*V_R、S_R*/

1. if l +1⩾r return
2. S_M←StringWithSmallestSize(S_L,S_R)
3. m←round((l+r)/2)
4. if size(S_L)< size(S_R)
5. then n←size(S_M) –size(S_L)
6. V_M←startV_L+n×Arr[l] startV_M←V_M–Arr[l]
7. else n←size(S_M)-size(S_R)
8. V_M←Arr[r]–startV_R+n×Arr[r] startV_M←Arr[r] end
9. Arr [m]←V_M
10. SubEncoding(Arr,l,m,startV_L,startV_M,S_L,S_M)
11. SubEncoding(Arr,m,r,startV_M,startV_R,S_M,S_R)

4 实验结果与分析

实验用向量进行 XML 区间编码。选用数据 Hamlet.xml[9] 测试更新过程中改进前后的向量编码空间占用。

Hamlet 共有元素节点 6 636 个，其根下有 5 个 act 元素节点，更新过程如下：首先删除第 1 个 act 子树，随即又恢复该子树，这是第 1 次更新。第 2 次更新在第 1 次更新的基础上删除第 2 个 act 子树，随即也恢复，按同样方式处理剩下 3 个 act 节点。每次更新后，文档和初始一样，但编码却发生变化，统计在每次更新后节点平均编码长度变化。

从图 2 可以看出，在更新前，改进前的向量编码和改进后的向量编码长度是一样的，但当删除又插入一棵子树时，改进前的向量编码需要较长编码位数，且随着更新次数的增多，编码长度增长较快，5 次更新后，编码长度增加一倍。而改进后，向量编码能利用到已删节点编码，使得编码位长较短。改进后的向量编码更适合于 XML 频繁更新。

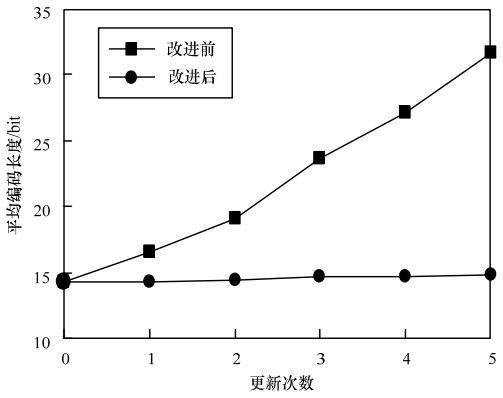

图 2 向量编码方法改进前后的平均编码长度对比

5 结束语

向量编码是一种将 XML 静态编码转化为动态编码的方法。为控制更新过程中向量编码的空间占用，本文对中间向量计算进行改进。在多项式时间内求出最短位长的中间向量，同时能大幅提高其批量分配的效率。实验结果验证了改进编码方法的有效性和可行性。

(下转第 78 页)

流传输速率的统计分布模型如表 6 所示，显示了对入链路和出链路流传输速率的拟合结果。

表 6 流传输速率的统计分布模型

类型	模型	D	置信水平
入链路	$0.84\mathrm{LN}_X(x;7.35,1.64)+0.16\mathrm{LN}_X(x;12.06,0.93)$	1.136	0.205
出链路	$0.52\mathrm{LN}_X(x;5.86,1.52)+0.48\mathrm{LN}_X(x;11.56,1.02)$	1.184	0.187

4.4 流长和流持续时间的关系

观察流长和流持续时间的关系。流长在这里指的是在一个流中所传输的字节量，而流持续时间被定义为流在路由器上的最后记录时间和开始记录时间的差值。通过计算流长和流持续时间的相关性，发现入链路流长和流持续时间的相关系数为 0.523，出链路则为 0.566。从这个结果中可以看出，在 2 条链路上流长和流持续时间并没有高度相关关系。画出在入链路上流长和流持续时间的散点图来观察数据的变化，即入链路流长和流持续时间的散点如图 7 所示，图 7 的横虚线和竖虚线分别是流长和流持续时间的平均值。从图 7 中可以观察到持续时间长的流所传输的数据范围在很大的范围内变化；持续时间短的流所传输流量的变化范围则要小一些；传输数据量大的流的持续时间大多较长，变化范围较小；而传输数据量小的流的持续时间一般不固定，且变化范围较大。也即流中同时出现了“mice 流”和“elephant 流”[7]以及“dragonfly 流”和“tortoise 流”现象的影响[8]。

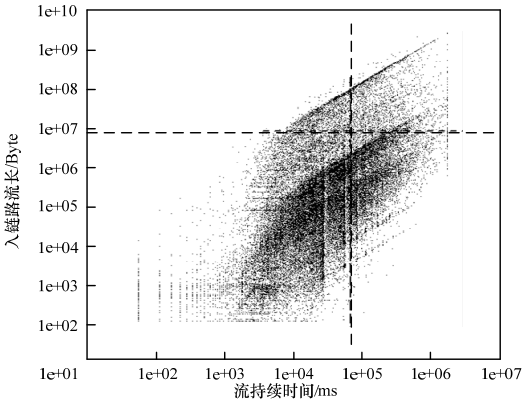

图 7 入链路流长和流持续时间的散点

5 结束语

本文通过统计分析的方法研究了Maze应用流量模型及其特点。分析结果表明，对数正态分布可以精确地表示模型流持续时间的分布，流长和流传输速率可以使用混合对数正态分布的方式拟合，因而可以有效地通过指导流量控制策略和流量工程手段来保证网络高效、可靠、稳定的运行。如何利用本文研究结果评估网络流量控制的效率将是今后的研究重点。

参考文献

[1] Vinh V T, Kensuke F, Hung N C, et al. A Study on P2P Traffic Characteristic Evaluation in the WIDE Backbone[C]//Proc. of the 6th International Conference on Information Technology and Applications. Hanoi, Vietnam: [s. n.], 2009.

[2] Satoshi O, Konosuke K. A Study on Traffic Characteristics Evaluation for a Pure P2P Application[C]//Proc. of the 16th Euromicro Conference on Parallel, Distributed and Network-based Processing. Toulouse, France: [s. n.], 2008.

[3] Han Y T, Park H S. Distinctive Traffic Characteristics of Pure and Game P2P Applications[C]//Proc. of the 10th International Conference on Advanced Communication Technology. Daejeon, Korea: [s. n.], 2008.

[4] 刘 刚, 方滨兴, 胡铭曾, 等. BitTorrent 流量的捕获方法及自相似性的评价[J]. 计算机应用研究, 2006, 23(5): 205-209.

[5] Azzouna N B, Guillemin F. Analysis of ADSL Traffic on an IP Backbone Link[C]//Proc. of Global Telecommunications Conference. San Francisco, USA: [s. n.], 2003.

[6] Stephens M A. EDF Statistics for Goodness of Fit and Some Comparisons[J]. Journal of the American Statistical Association, 1974, 69(347): 730-737.

[7] Papagiannaki K, Taft N, Bhattacharyya S, et al, A Pragmatic Definition of Elephants in Internet Backbone Traffic[C]//Proc. of the ACM SIGCOMM Internet Measurement Workshop. New York, USA: ACM Press, 2002.

[8] Nevil B, Claffy K C. Understanding Internet Traffic Streams: Dragonflies and Tortoises[J]. IEEE Communications Magazine, 2002, 40(10): 110-117.

编辑 刘 冰

(上接第 66 页)

参考文献

[1] 孟小峰. XML 数据管理: 概念与技术[M]. 北京: 清华大学出版社, 2009: 58-75.

[2] Tatarnov I, Viglas S D, Beryer K, et al. Storing and Querying Ordered XML Using a Relational Database System[C]//Proc. of 2002 ACM SIGMOD International Conference on Management of Data. New York, USA: ACM Press, 2002: 204-215.

[3] Zhang Chun, Naughton J F, DeWitt D J, et al. On Supporting Containment Queries in Relational Database Management Systems[C]//Proc. of 2001 ACM SIGMOD International Conference on Management of Data. New York, USA: ACM Press, 2001: 425-256.

[4] Li Changqing, Ling T W, Hu Min. Efficient Update in Dynamic XML Data: From Binary String to Quaternary String[J]. The VLDB Journal, 2008, 17(3): 573-601.

[5] Li Changqing, Ling T W. QED: A Novel Quaternary Encoding to Completely Avoid Re-labeling in XML Updates[C]//Proc. of the 14th ACM International Conference on Information and Knowledge Management. New York, USA: ACM Press, 2005: 501-508.

[6] Xu Liang, Bao Zhifeng, Ling T W. A Dynamic Labeling Scheme Using Vectors[C]//Proc. of the 18th International Conference on Database and Expert Systems Applications. Berlin, Germany: Springer-Verlag, 2007: 130-140.

[7] Xu Liang, Ling T W, Wu Huayu, et al. DDE: From Dewey to a Fully Dynamic XML Labeling Scheme[C]//Proc. of SIGMOD'09. New York, USA: ACM Press, 2009: 719-730.

[8] Graham R L, Knuth D E, Patashnik O. Concrete Mathematics: A Foundation for Computer Science[M]. [S. l.]: Addison-Wesley Publishing Company, 1989: 115-137.

[9] NIAGARA Experimental Data[EB/OL]. (2010-08-21). http://www.cs.wisc.edu/niagara/data.html.

编辑 陆燕菲